

ALGORITMOS Y ESTRUCTURAS DE DATOS III - 1er Recuperatorio

Fecha examen: 12-JUL-2019 / Fecha notas: a determinar

Completar:	Nº Orden	Apellido y nombre	L.U.	Cant. hojas ¹
	29	FESTINI SANTIAGO	311/17	7
No completar:	Nota (Nº)	Nota (Letras)	Docente	
	9,8	NUEVE CON 80/100		

1. Un grafo funcional es un (pseudo) grafo dirigido en el cual todo vértice v cumple que $d_{out}(v) = 1$, de modo tal que representa a alguna función de un conjunto en sí mismo. 2.5 p

Sea G un grafo funcional. Demostrar que existe un vértice v tal que $d_{in}(v) = 0$ si y sólo si existe un vértice w tal que $d_{in}(w) \geq 2$.

2. Sea s una cadena de caracteres de longitud n . Sea d (diccionario) una lista de p cadenas de caracteres (palabras) distintas entre sí, cada una de las cuales tiene longitud no nula y $O(1)$. Diseñar un algoritmo que determine la cantidad de maneras distintas de partir s en subcadenas que están en d . Ejemplos: 2.5 p

- Para $s = \text{"adorador"}$ y $d = (\text{"ador"}, \text{"noseusa"})$ el resultado es 1.
- Para $s = \text{"adorador"}$ y $d = (\text{"adora"}, \text{"dorador"})$ el resultado es 0.
- Para $s = \text{"abracadabra"}$ y $d = (\text{"a"}, \text{"abra"}, \text{"bra"}, \text{"cad"}, \text{"cada"})$ el resultado es 6 ("a-bra-cad-a-bra" , "a-bra-cad-abra" , "a-bra-cada-bra" , "abra-cad-a-bra" , "abra-cad-abra" , y "abra-cada-bra").
- Para s formada por n repeticiones del carácter "a" y $d = (\text{"a"}, \text{"aa"})$ el resultado es el n -ésimo número de Fibonacci.

El algoritmo debe tener complejidad $O(np)$ y estar basado en programación dinámica. Mostrar que el algoritmo propuesto es correcto y determinar su complejidad. Justificar.

SUGERENCIA: Resolver para cada prefijo de s .

3. En el Torneo Tricolor compiten 3 equipos: azul, amarillo y rojo. Las reglas del torneo aparecen a continuación. 2.5 p

- No debe haber amigos dentro del equipo azul.
- No debe haber amigos dentro del equipo amarillo.
- Los miembros del equipo rojo deben ser amigos de todos los que participan en el torneo.
- Cada equipo debe tener al menos un miembro.

Diseñar un algoritmo eficiente basado en grafos que dada la lista de amigos dentro de un grupo de personas, decida si es posible armar con ellas los 3 equipos para el Torneo Tricolor. La entrada del algoritmo es la cantidad p de personas, la cantidad r de relaciones de amistad entre ellas, y para cada amistad las dos personas que son amigas (identificadas por enteros entre 1 y p). La relación de amistad se supone simétrica. Mostrar que el algoritmo propuesto es correcto y determinar su complejidad. Justificar. El mejor algoritmo que conocemos tiene complejidad $O(p+r)$, lo cual es necesario para obtener puntaje máximo en este ejercicio.

SUGERENCIA: Resolver primero sin tener en cuenta la última regla.

4. Un grafo (simple) se dice 1-árbol si es un árbol con un eje agregado. 2.5 p

Sea G un grafo conexo con pesos asociados a sus ejes, y que tiene al menos un ciclo. Un 1-árbol generador de G es un 1-árbol que es subgrafo generador de G . Un 1-árbol generador mínimo de G es 1-árbol generador de G que tiene peso mínimo en el conjunto de los 1-árboles generadores de G .

Sea H un 1-árbol generador mínimo de G , y sea f un eje de peso máximo en el único ciclo de H . Demostrar que $H - f$ es un árbol generador mínimo de G .

SUGERENCIA: Demostrar que el peso de $H - f$ es menor o igual que el peso de cualquier árbol generador de G .

② N.º. 29 FESJINI SANTIAGO 311/37 2,5p

$$1. \text{Dout}(v) = I \quad \forall v \in V(G)$$

QUA $\exists w$ VÉRTICE $v / \text{Din}(v) = 0 \Leftrightarrow \exists w$ VÉRTICE $w / \text{Din}(w) \geq 2$.

POR PROPIEDAD SE QUE:

$$M = \sum_{v \in V(G)} \text{Dout}(v)$$

$n = n \cdot I = n$ YA QUE TODO VÉRTICE $\in V(G)$ TIENE GRADO DE SALIDA I POR G SER GRADO FUNCIONAL.

POR OTRA PROPIEDAD SE QUE

$$\sum_{v \in V(G)} \text{Dout}(v) = \sum_{v \in V(G)} \text{Din}(v) = n \cdot I$$

② ~~$\forall v \in V(G) / \text{Din}(v) = 0 \Rightarrow \exists w \in V(G) / \text{Din}(w) \geq 2$~~

SUP. QUE NO, ENTONCES ~~$\exists w \in V(G) / \text{Din}(w) \geq 2$~~

$$\Rightarrow \forall w \in V(G), \text{Din}(w) < 2$$

$$\Rightarrow \forall w \in V(G), \text{Din}(w) \leq 1$$

YA QUE LOS GRADOS DE LOS VÉRTICES SON VALORES $\in \mathbb{N}_0$

$$\Rightarrow \sum_{v \in V(G)} \text{Din}(v) = \sum_{v \in V(G)} \text{Din}(v) + \sum_{w \in V(G) \wedge w \neq v} \text{Din}(w) \leq I$$

// $\{w \in V(G) \wedge w \neq v\} \rightarrow w = n-1$
PORQUE LO CUENTO POR SEPARADO

$$\Rightarrow \sum_{v \in V(G)} \text{Din}(v) = 0 + \sum_{w \in V(G) \wedge w \neq v} \text{Din}(w) \leq 0 + (n-1) \cdot I = n-I$$

// POR $\ast I$

$$\Rightarrow n \leq n-I \quad \text{ABS!} \quad \checkmark$$

$$\Rightarrow \exists v \in V(G) / \text{Din}(v) = 0 \Rightarrow \exists w \in V(G) / \text{Din}(w) \geq 2$$

$\Leftarrow \exists w \in V(G) / \text{din}(w) \geq 2 \Rightarrow \exists v \in V(G) / \text{din}(v) = 0.$

SUPONGO QUE NO, ENTONCES $\neg \exists v \in V(G) / \text{din}(v) = 0$

$\Rightarrow \forall v \in V(G), \text{din}(v) \neq 0$

$\Rightarrow \forall v \in V(G), \text{din}(v) \geq 1$

$\Rightarrow \forall v \in V(G), \text{din}(v) \geq 1$

ELIMINAMOS EL CASO $\text{din}(v) < 0$ PORQUE
 LOS VALORES DE LOS GRADOS $\in \mathbb{N}_0$

$\Rightarrow \sum \text{din } V(G) = \overbrace{\text{din}(w)}^{\geq 2} + \sum_{\substack{v \in V(G) \wedge v \neq w \\ \text{IGUAL QUE EN EL CASO ANTERIOR, SE M\u00c1S INFORMACI\u00d3N ACERCA DE W CON RESPECTO AL RESTO DE LOS V\u00c9RTICES POR LO QUE DEBE SER SEPARADO.}}}^{\geq 1} \geq 2 + (n-1) \cdot 1 = n+1$

//
 n por *1

IGUAL QUE EN EL CASO ANTERIOR, SE M\u00c1S INFORMACI\u00d3N ACERCA DE W CON RESPECTO AL RESTO DE LOS V\u00c9RTICES POR LO QUE DEBE SER SEPARADO. ~~NO EXCLUIDO DE LA SUMATORIA~~ Y LO CONTEMPO SEPARADO.

$\Rightarrow n \geq n+1$ ✓

ABS!, $\Rightarrow \exists w \in V(G) / \text{din}(w) \geq 2 \Rightarrow \exists v \in V(G) / \text{din}(v) = 0.$

(3) N.O. 29

FESTINI SANTIAGO

311/17

(2,5)

$$2. \quad f(i, K) = \begin{cases} 1 & \text{Si } i \geq \text{TAM}(S) = n \quad \checkmark \\ 0 & \text{Si } K \geq \text{TAM}(D) = p \quad \checkmark \\ f(i + \text{D}[K], 0) & \text{Si } \text{D}[K] \text{ ES PREFIJO DE } \underline{S[i:n]} \\ + f(i, K+1) & \\ f(i, K+1) & \text{SINO.} \quad \checkmark \end{cases}$$

~~f ES LA FUNCIÓN RECURSIVA TAL QUE~~ $f(i, K)$ EQUIVALE A LA CANTIDAD DE FORMAS EN QUE SE PUEDE PARTIR UNA CADENA DE CARÁCTERES S , A PARTIR DE SU CARÁCTER i EN p SUBCADEJAS EXISTENTES EN D Y CUYA PRIMER CADENA ESTA LIMITADA A SER A PARTIR DE LA CADENA NÚMERO K . \checkmark

LLAMANDO A f DESDE ~~$i=0$~~ $i=0$ Y $K=0$ CONTABILIZAMOS TODAS LAS POSIBLES COMBINACIONES DE SUBCADEJAS EXISTENTES EN D CON LAS QUE PODEMOS FORMAR S . \checkmark

NOTAR QUE DE ESTA MANERA, PODEMOS USAR COMO ESTRUCTURA DE MEMORIZACIÓN, UN SIMPLE VECTOR/DICCIONARIO DE TAMAÑO n , LO QUE DORBA AL ALGORITMO COMPLEJIDAD ESPACIAL $O(n)$. ESTO SOLO ES ALCANZABLE GRACIAS A LA MANERA EN QUE EFECTUAMOS LA RECURSIÓN, ANALIZANDO PRIMERO EL CASO EN QUE USAMOS A UN PREFIJO VÁLIDO COMO PREFIJO Y LUEGO EL CASO EN QUE DECIDIMOS IGNORARLO Y NO USARLO. EL DICCIONARIO MEMORIZADOR ALMACENA EN SU POSICIÓN $0 \leq i < n$ LAS MANERAS DE FORMAR EL SUBSTRING ~~$S[i:n]$~~ $S[i:n]$ A PARTIR DE CADENAS EXISTENTES EN D . \checkmark Y COMO AVANZO FRECUENTEMENTE SOBRE i Y NO RETROCEDO HASTA COMPLETAR UNA PASADA DE LAS

PALABRAS A PARTIR DEL CARACTER i , ENTONCES NUNCA ESTOY USANDO MEMORIZACIONES A MENAS, INVÁLIDAS, NI ~~REPRIMIENDO~~ REPRIMIENDO AUN OPORTUNIDADES DE REUSAR COSAS YA CALCULADAS. Perfecto.

VECTOR CINTO MEMO (n , \bullet); ^{→ TAMAÑO} ^{→ VALOR INICIAL = UNDEFINED.} (me gusta simbolizaste indefinido con un tachón XD)
 STRING S / VECTOR (STRING) D; // DICC. DE MEMORIZACIÓN GLOBAL. es peticos.
 // STRING DATO. Y LOS 5 PALABRAS DATO.

```

int V(int i, int k) {
    if (i == TAN(S)) return 1;
    if (k == TAN(D)) return 0;
    if (k != 0) {
        if (MEMO[i] != UNDEFINED) return MEMO[i];
        else if (MEMO[i] == UNDEFINED) return MEMO[i];
        if (ESPREFISO?(S[i,n], D[k])) return V(i+ID[k], 0) + V(i, k+1);
        else return V(i, k+1);
    }
    // K = 0! TENEMOS QUE MEMORIZAR.
    if (MEMO[i] != UNDEFINED) return MEMO[i];
    else // TENEMOS QUE DEFINIR MEMO! (CON EL RESULTADO QUE VAYAMOS A CALCULAR.
        if (ESPREFISO?(S[i,n], D[k]))
            MEMO[i] = V(i+ID[k], 0) + V(i, k+1);
        else
            MEMO[i] = V(i, k+1);
    return MEMO[i];
}
    
```

~~no estábamos definiendo nada~~

~~no estábamos definiendo el sentido de este if.~~

// RETORNO DE MEMORIZAR

~~if (MEMO[i] != UNDEFINED) return MEMO[i];~~
~~else if (MEMO[i] == UNDEFINED) return MEMO[i];~~

if (ESPREFISO?(S[i,n], D[k])) return V(i+ID[k], 0) + V(i, k+1);
 else return V(i, k+1);

else // K = 0! TENEMOS QUE MEMORIZAR.

if (MEMO[i] != UNDEFINED) return MEMO[i];

else // TENEMOS QUE DEFINIR MEMO! (CON EL RESULTADO QUE VAYAMOS A CALCULAR.

if (ESPREFISO?(S[i,n], D[k]))
 MEMO[i] = V(i+ID[k], 0) + V(i, k+1);

else
 MEMO[i] = V(i, k+1);

return MEMO[i];

(4)

N.O. 29

FESTINI SANTIAGO 31/17

HOJA N°

FECHA

CONT. ES. 2.

COMO TODAS LAS OPERACIONES REALIZADAS EN EL ALGORITMO SON $O(1)$ EN COMPLEJIDAD TEMPORAL Y GRACIAS A NUESTRA ESTRUCTURA DE MEMORIZACIÓN NOS ASEGURAMOS DE UNICAMENTE RECORRER TODAS LAS PALABRAS DEL CONJUNTO $D, (P)$ UNA VEZ POR CADA CARACTER DEL STRING S (TAMAÑO n), LA COMPLEJIDAD TEMPORAL NOS QUEDA $O(np.)$ ✓

POR ÚLTIMO, VEAMOS BREVEMENTE QUE ES CORRECTO:

~~~ SI NOS EXCEDEMOS EL TAMAÑO DEL STRING ENTONCES LO LOGRAMOS COMPLETAR POR LO QUE DEVOLVEMOS 1.~~

✓ - SI SOBREPASAMOS EL TAMAÑO DEL STRING ENTONCES LO LOGRAMOS COMPLETAR POR LO QUE DEVOLVEMOS 1.

✓ - SI NOS EXCEDEMOS DE CANT. DE PALABRAS ENTONCES YA RENUNCIAMOS TODAS Y NO PODEMOS SEGUIR PROBANDO CON OTRAS,  $\Rightarrow$  DEVOLVEMOS 0.

✓ - Y SI NO, EN CASO DE SER PREFISOS CUALQUIERA AMBOS CASOS, AQUEL EN QUE SE USA Y AQUEL EN QUE NO Y LOS SUMA.

✓ - Y POR ÚLTIMO, SI NO ES PREFISO NO SE CUENTA Y SIGUE CON LA RECURSIÓN.

✓ - COMO LOS PARÁMETROS DE LA FUNCIÓN SON CRECIENTES NO REPETIMOS CASOS!

✓ - ACLARACIÓN: LA FUNCIÓN ES PREFISO? ES  $O(1)$  PORQUE TODA PALABRA  $\exists P$  ES  $O(1)$  EN ESPACIO.

### 3- TORNEO TRICOLOR → TRES EQUIPOS

- AZUL : NO AMIGOS ENTRE SÍ.
- AMARILLO : NO AMIGOS ENTRE SÍ.
- ROJO : AMIGOS DE TODOS LOS PARTICIPANTES.

IDEA 8 • RECORDAR LAS RELACIONES Y SEPARAR LAS PERSONAS QUE SON AMIGOS DE TODOS Y LAS MANDO AL EQUIPO ROJO.  
• ME FISO SI LOS RESTANTES LOS PUEDO SEPARAR EN DOS CONJUNTOS DE ANTIPATÍA (EQUIPOS ~~ROJO~~ AZUL Y AMARILLO), DONDE UN CONJUNTO DE ANTIPATÍA ES AQUEL QUE CUMPLE QUE:  
 $\forall P, P' \text{ PERSONAS} \in C \Rightarrow P \text{ Y } P' \text{ NO SE RELACIONAN ENTRE SÍ.}$

ENTONCES Y DESPUÉS PULIMOS CASOS BORDE:  $|RELACIONES| = R$

```

// # PERSONAS # RELACIONES
// LISTA DE RELACIONES SIMÉTRICAS
T. T. (INT P, INT R, VECTOR<INT, INT> RELACIONES) {
    INT PARTROSO [P]; PARTROSO ← 0; // VALOR AL ARRAY DE TAMAÑO P O(P)
    PARA (i : 0 ≤ i < R) {
        PARTROSO [V(RELACIONES[i])] ++; // CUÉNTA CANTIDAD DE RELACIONES
        PARTROSO [W(RELACIONES[i])] ++; // DE CADA PERSONA.
    }
    GRAPH G = (V = {1, ..., P}, E = ∅) // CREO UN GRAFO CUYOS VÉRTICES O(P)
    PARA (i : 0 ≤ i < R) { // SON PERSONAS, USO LISTA DE ADY.
        REL = RELACIONES[i]
        SI PARTROSO[V(REL)] ≠ P-1 ∧ PARTROSO[W(REL)] ≠ P-1; // VÉRTICE
            AGREGO EN G A W COMO VECINO DE V; // SE RELACIONA
            AGREGO EN G A V COMO VECINO DE W; // CON TODOS ⇒
            // LO DEJO EN EQ. ROJO
        }
    }
    // Y NO LO PASO AL GRAFO QUE
    // VDT A USAR PARA REVISAR BIPARTITUD.
}

```

\*FIN  
CREACIÓN G



// USO DFS MARCA SOBRE TODOS LOS VÉRTICES DE  $G$ . LO EXPLICO MEJOR ABAJO

$O(P)$  PARA  $(i : 1 \leq i \leq |V(G)|)$   $\rightarrow$  ACOTADO POR  $P$   
DFS MARCA  $(G, i, \text{AZUL})$   
 $O(R)$  PARA  $(i : 0 \leq i < |E(G)|)$   $\rightarrow$  ACOTADO POR  $R$ .  
SEA  $e_i$  LA  $i$ -ÉSIMA ARISTA;  
SI  $(\text{COLOR}(V(e_i)) = \text{COLOR}(W(e_i)))$   
RETURN FALSE;  
RETURN TRUE;  
RECORRO TODAS LAS ARISTAS ENTRE PERSONAS DE LOS EQUIPOS AZUL Y AMARILLO Y REVISO QUE NO HAYA RELACIÓN ENTRE DOS DEL MISMO EQUIPO.

DFS MARCA, RELIBE UN GRAFO, UN VÉRTICE Y UN COLOR, AZUL O AMARILLO, FUNCIONA COMO EL DFS VISTO EN LA TEORÍA PERO QUE ALTERNAR EL COLOR CON QUE PINTA Y ~~NO HACER NADA~~ SI EL VÉRTICE YA ESTABA PINTADO NO HACE NADA.

IDEA MÁS DETALLADA... BUSCO POR VÉRTICES UNIVERSALES, A ELLOS LES PUEDO ASIGNAR EL COLOR ROJO. EN LOS VÉRTICES RESTANTES USO DFS PARA ANALIZAR BIPARTITO, SI LOGRO TODO ESTO  $\Rightarrow$  PUEDO SEPARARLOS EN EQUIPOS (VIOLANDO LAS RESTRICCIONES).

FALTAN LIBROS USOS BORDE:

1º: ~~MADE~~  $P$  DEBE SER  $\geq 3$  YA QUE SINO ALGÚN EQUIPO SE QUEDA VAGO.

2º: UNA VEZ CREADO  $G$  (\*FIN RELACIÓN  $G$ ), SI  $P = |V(G)|$  ENTONCES DEVOLVER FALSO YA QUE NO HAY PERSONAS EN EL EQUIPO ROJO.

\*~~25.2~~  $|V(G)| \geq 3$



6

N.O. 29

FBSSINI SANTIAGO 25/1/17

HUJATE

FECHA

Cont. 3 y, si  $|V(G)| = 0 \Rightarrow$  TODOS ERAN VERTICES

UNIVERSALES, ~~PARABEROS QUE HAY~~ POR LO QUE PODEROS  
SEPARAR 2 CUALESQUIERA Y ASIGNAR 1 AL EQ. AZUL Y OTRO  
AL EQ. AMARILLO Y GANANDOS!

NOTAR QUE  $\bigcirc - \bigcirc$  ES BIPARTITO

ESTO SOLO VALE SI ANTES  
MOS ASEGURAMOS QUE  
 $P \geq 3$  COMO DÍJIMOS.

POR ÚLTIMO, LA COMPLESIÓN ES  $O(P+R)$  COMO ERA REQUERIDO.  
ESTO ES OBSERVABLE POR LAS ADAPTACIONES AL COSTADO DEL  
ALGORITMO.

FERRINI SANTIAGO 31 I / 27

H I - ÁRBOL GENERADOR <sup>mínimo</sup> DE 6

✓ ESE DE PESO MÁXIMO EN EL ÚNICO CICLO DE  $N$

$\Rightarrow H - v \in S \text{ A.6. } \pi. \text{ O} \in G$

SEA  $\gamma$  UN ÁRBOL GENERADOR DE  $G$  CUALQUIERA.

QUA  $P(W) - P(V) \leq P(T)$  DONDE  $P(\cdot)$  REPRESENTA LA FUNCIÓN DE PESO.

PARTIENDO DE QUE:

\*1 [  $P(H) \leq P(Y) + P(Z)$  ]  $Z \in E(6) \wedge Z \notin E(7)$

SEA  $z$  ~~UN~~ <sup>UN</sup> EJE QUE EXISTE EN EL CICLO DE  $H$  Y NO ESTA EN  $T$ . SABEMOS QUE SIEMPRE EXISTE AL MENOS UNA YA QUE SI NO  $T$  TENDRIA UN CICLO, PERO  $T$  ES ÁRBOL  $\Rightarrow$  ABS!

~~WILLIAM PERMAN~~ ~~CHAS. W. PERMAN~~

~~XXXXXXXXXXXX~~ U6A05

$$P(H) - P(V) \leq P(T) + P(Z) - P(Z)$$

$$P(H) \leq P(Y) + P(Z) \text{ POR } *1 \text{ (HIPÓTESIS)}$$

^

$$-P(V) \in -P(Z), \quad P(Z) \subseteq P(V)$$

$(2) \in P(K)$   
VALE PORQUE AMBAS SON ARISTAS  
PERTENECIENTES AL CICLO DE  $H$   
Y SABEMOS POR HIPÓTESIS  
QUE  $H$  ES LA MAYOR.



VALE ENTON CES:

$$P(H) - P(V) \leq P(T) + \cancel{P(Z)} - P(Z)$$

$$\Rightarrow P(H) - P(V) \leq P(T) + \cancel{P(Z)} - \cancel{P(Z)}$$

$$\Rightarrow P(H) - P(V) \leq P(T)$$

QUE ES LO QUE QUERÍAMOS VER.

OK. Faltaba argumentar que  $H-f$  es un AG de  $G$ . (Alumnos iniciados)

[Z.S.] Francisco.